A Parallel Learning Automata Method of the Graph Partitioning Problems

Shigeya IKEBOU, Jijun WU, Yue ZHAO, Fei QIAN
Faculty of Engineering, Hiroshima Kokusai Gakuin University
6-20-1 Nakano Aki-ku, Hiroshima, 739-0321 JAPAN

In this paper, we have presented the learning automata solution to the NP-hard problem, the graph-partitioning problems (GPP), which involves partitioning the nodes of a graph G into K sets of equal size so as to minimize the sum of the costs of the edges having endpoints in different sets. We proposed a method for this problem, a parallel learning automata (LA) method. This method aimed at the parallel and solution accuracy of the GPP and applied local evaluation function. The simulation results show that, our learning automata based method has good performance on computing time and has some superiority with the increase in the number of nodes.

1. Introduction

Combinatorial optimization problem is one of the important fields in modern engineering. It is mostly related to pattern recognition, graphics processing, layout synthesis in VLSI design, manufacture scheduling and artificial intelligence etc. For a common optimization problem, there are several local optimal solutions. Therefore in order not to stop at these local optimal solutions, it's necessary to develop new methods for finding the optimal solution in the whole field. Also it's known that when the size of the problem gets bigger, the number of possible solutions increases exponentially, and it's difficult to find the optimal solution. The common evaluation of a computing method for a combinatorial optimization problem is based on the computing complexity and convergence etc. Convergence relies on the complexity, so it can be easily influenced by the evaluation function, which evaluates the quality of a solution of the computing method. Various computing methods have been suggested to get high quality solution with fast speed for this kind of problem[1]-[4].

In present paper, as one of the combinatorial optimization problem, graph-partitioning problems (GPP) is considered and provided with a parallel learning automaton (LA) method. This method aimed at the parallel and solution accuracy of the GPP and applied local evaluation function. To verify the effectiveness of this computing method, MIP (Message Passing Interface), a standard way of message passing is used on parallel computers in discrete memory environment and executed on cluster.

2. Learning automaton computing model

For a graph $G = (N, E)$ defined by a node set $N$ and link set $E$, GPP is the problem of partitioning graph $G$ into $K$ sub-graph under the following conditions $(G = \{G_j = (N_j, E_j) | j = 1, 2, \cdots, K\})$.

(a) Minimize the sum of costs between sub-graphs.

(b) Equalize the number of nodes in the sub-graphs if possible.

(c) Assign one node to only one sub-graph.

Under these limiting conditions, the evaluation function is generally defined in following way:

General evaluation function:
\[ f(t) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} c_i \left(1 - s_{ik}\right)s_{ik} - P \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \left(1 - s_{ik}\right)s_{ik} + Q \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} s_{ik}s_{jm} \]  \hspace{1cm} (1)

where,

\( P, Q \): the parameter expressing the reliance to the limiting conditions.

\( c_i \): link cost between node \( n_i \) and \( n_j \).

\( s_{ij} \) = \{0,1\}, \( i \in n, 1 \leq j \leq K \).

0: not belong to \( G_j \), 1: belong to \( G_j \).

Each item in equation (1) corresponds to each of the limiting conditions respectively.

Learning automaton is defined as the stochastic automaton (SA) working in a random environment (RE) [1].

To apply LA to the GPP,

1. GPP's "changes in evaluation function value" are taken as LA's "random environments",
2. one SA is assigned to each node on the graph,
3. each node monitors the division of information through SA and in the mean time predicts the assignment to its own sub-graph.

In this paper, the learning automata team model shown in Fig.1 is applied.

The learning automaton team consists of \( n \) stochastic automaton \( SA^1, SA^2, \cdots, SA^n \), and the strategy set taken by each automaton is formed from the output set of each stochastic automaton. Also each time \( SA^i \) taking strategy \( j \) is presented as \( y'(i) = y'_j \). For each automaton, the target of this team performance is to find out a mix-output \( (y'_1, y'_2, \cdots, y'_n) \) so that it minimizes the average penalty given by the random environment.

Learning automaton’s team model is in the category of team type non-cooperative game. It has been verified that there are Nash’s fixed points (optimal mixing strategy) in the strategy space [2].

3. Parallel learning automaton method for GPP

3.1. Local evaluation function for GPP

GPP’s general evaluation function computes the cost changes for the entire graph, therefore the computing time is long and parallel treatment is difficult. Here the local evaluation function for GPP only targets and evaluates the cost between two nodes.

If node \( n_i \) and \( n_j \) are the target nodes, the local evaluation function for the GPP can be presented as following.

Local evaluation function \( f : \)

\[ f = \sum_{a \in R_i} c_a + \sum_{n \in N_i} c_{n} \]

\( c_a \): Link cost between node \( n_i, n_{j} \).

\( N_i \): \( n_i \)'s adjacent node set that each element is in sub-graph \( N_j \), i.e.

\[ N_i = \{ n | n \in N_i, n \in N_j, \exists (n_i, n_j) \in E \} \]

\( N_j \): \( n_j \)'s adjacent node set that each element is in sub-graph \( N_j \), i.e.

\[ N_j = \{ n | n \in N_i, n \in N_j, \exists (n_i, n_j) \in E \} \]

Where condition (b) and (c) are involved in the initial state. Like other local search method, this method also relies on the initial state, so there is the possibility to be trapped in the local optimal solution.

When applying the local evaluation function for GPP, the evaluation is like following.
Input signals: $r = \{0, 1\}$.

$y = \{y_l | j = 1, 2, \cdots, K\}$

$\Phi = \{\phi_j | j = 1, 2, \cdots, K\}$

$P'$: output selection probability distribution

$\Phi = \{\phi_j | j = 1, 2, \cdots, K\}$

$Y'$: output set

$\Phi$: state set

$pt = (p_t, p_{L}, p_k)$

$0'$: output function

$T'$: reinforcement scheme

Random environment is defined as following:

$RE = \{Y, R, f\}$

$Y$: input set, $Y = \{Y^i | i = 1, 2, \cdots, n\}$.

$R$: output (reinforcement signal) set, $R = \{r^i | i = 1, 2, \cdots, n\}$.

$f$: local evaluation function.

Here output set $Y'$'s each output $y^i$ corresponds to each assigned sub-graph $G_j$, and the output $y^j(t)$ means node $n_i$ is assigned to the $j$th sub-graph $G_j$.

At time $t$, when node $i$ is assigned to $G_k \subset G$, the state probability vector $p^i(t)$ is modified with following linear reward-inaction[1].

$$r^i = 0; /* reward */$$

$$p^i_m(t+1) = (1-\theta)p^i_m(t) + \theta$$

$$m \neq k$$

$$r^i = 1; /* penalty */$$

$$p^i_m(t+1) = (1-\theta)p^i_m(t)$$

$$m = 1, 2, \cdots, K$$

$\theta$ is the learning parameter and $0 < \theta < 1$.

PARAMETERS

- node number $n$, division number $K$.
- Set the learning parameter $\theta$.

ALGORITHM

Initialization:

- Randomly partition the node set $N$ into same size.
- Assign stochastic automaton $SA^i$, $(i = 1, 2, \cdots, n)$ to each node $n_i$.
- Let $SA^i$'s output selection probability distribu-
tion is \( p'_i(0) = 1/K \).

**METHOD**

**Step 1:** Randomly decide \( K/2 \) pairs from the entire sub-graphs.

(Step 2-7 are executed parallely to each pair)

**Step 2:** Randomly selects two \( SA', SA'' \) \((G'_m \neq G''_m)\).

**Step 3:** For \( SA', SA'' \), determine the output \( y'_i, y''_m \) according to their output selection probability distribution \( p'(t), p''(t) \).

**Step 4:** For node \( n_i \in G_m, n_j \in G_m \), compute present cost \( (f_{old\_cost}) \) with Eq. (1)

**Step 5:** Replace the nodes, compute and evaluate the new cost \( (f_{new\_cost}) \) with Eq. (1)

**Step 6:** \( r^{i,j} = \begin{cases} 0 & \text{reward, if } f_{new\_cost} \leq f_{old\_cost} \\ 1 & \text{penalty, otherwise} \end{cases} \) \( (6) \)

**Step 7:** If penalty, node \( n_i, n_j \) are returned to the former subset.

**Step 8:** Update the state probability vector \( p'(t), p''(t) \) according to the reinforcement scheme \( T', T'' \).

**Step 9:** Return to Step 1 when finished for all the pairs.

Along with this repeated execution, the optimal output with maximum reward is determined. This optimal output is equivalent to the optimum assignment of GPP.

4. Experimental Verification

To verify the proposed method, computer experiment is done on a geometric graph \( U_n (n = [200, 500]) \) shown in Fig. 3. The graph \( U_n \) is a random graph with \( n \) nodes, each node is randomly placed into a quadratic space of \( [0, 1] \), and the nodes randomly link to each other within a square of certain length. Division number \( K = \{10, 20, 50, 100\} \).

As a comparison, the learning automaton computing method with local evaluation function for GPP and the simulated annealing method [3][4] are dealt with. The experimental parameter and finishing condition for each method are as following.

**EXPERIMENT 1:**

1. Learning automaton computing method with local evaluation function for GPP: LA
   - Learning parameter: \( \theta = 0.003 \), repetitions: 100000.
   - Finishing condition: repeat till the arbitrary repetition number.

2. Parallel learning automaton computing method with local evaluation function for GPP
   - Learning parameter: \( \theta = 0.003 \), repetitions: 100000.
   - Finishing condition (a):
     - repeat till the arbitrary repetition number. : PLA
   - Finishing condition (b):
     - the cost changes continue for 500 times, no more than 0.000001. : PLA'

3. Simulated annealing method: SA
   - Initial temperature: 20.0, finishing temperature: 0.01, repetition at each temperature: 10000, annealing coefficient for constant annealing method:0.95.
   - Finishing condition: repeat till any arbitrary finishing temperature.
EXPERIMENT 2: The performance of the proposed method on PC-cluster system with MPI

Experimental Environment:
Nodes: 8
CPU: Pentium Celeron 400
RAM: 128M
HDD: 10G
NIC: Myrinet M2F-PC132C
Myrinet Switch M2M-DUAL-SW-16

20 trials for each method, the results of average values are shown in Fig. 5-8. Fig 5 is the cost change comparison between the learning automaton computing method with local evaluation function for GPP and the proposed method (under finishing condition (a)). Fig. 6 is the comparison of preciseness. Fig 7, 8 show the computing time to reach the preciseness in Fig. 6.

![Cost evolution of LA and parallel LA](image)

**Fig.5 Cost evolution of LA and parallel LA (K = 50)**

It can be seen from Fig.5 that compared to the learning automaton computing method with local evaluation function for GPP, it's paralleled version of our proposed method has a faster convergent speed. This can be reasoned by the increase in the selected chances of each node (stochastic automaton).

From Fig.6 it can be seen that the solution preciseness of learning automaton computing method with local evaluation function for GPP drops with the increase of node number. But the proposed method can provide solution preciseness very close to that of simulated annealing with the increase of node number.

From Fig. 7, 8, it's known that with the proposed method under finishing condition (a), because of the increase in the division number, the computing time is greatly increased compared to the learning automaton with local evaluation function for GPP. This is because the computing is done at each step for $K/2$ pair of sub-graphs. On the other hand, at finishing condition (b), a very good result is obtained.

| Table 1 Simulation results (costs for N200). |
|---|---|---|---|---|
| N200 | LA | PLA | PLA' | SA |
| 10 Min | 19.4 | 18.2 | 19.0 | 19.0 |
| Mean | 20.8 | 19.5 | 20.6 | 21.2 |
| Max | 23.9 | 19.9 | 24.9 | 23.9 |
| 20 Min | 36.1 | 34.6 | 35.3 | 34.7 |
| Mean | 37.5 | 36.2 | 36.6 | 36.3 |
| Max | 39.4 | 37.6 | 37.7 | 38.5 |
| 50 Min | 59.3 | 59.3 | 59.2 | 58.9 |
| Mean | 60.1 | 59.6 | 59.7 | 59.7 |
| Max | 60.7 | 60.1 | 60.3 | 60.2 |
| 100 Min | 72.1 | 72.1 | 72.1 | 72.0 |
| Mean | 72.3 | 72.2 | 72.3 | 72.3 |
| Max | 72.5 | 72.3 | 72.5 | 72.5 |

| Table 2 Simulation results (costs for N500). |
|---|---|---|---|---|
| N500 | LA | PLA | PLA' | SA |
| 10 Min | 57.4 | 48.0 | 54.2 | 48.8 |
| Mean | 65.3 | 53.7 | 62.9 | 56.8 |
| Max | 75.1 | 60.2 | 58.3 | 67.1 |
| 20 Min | 89.2 | 70.3 | 74.6 | 71.4 |
| Mean | 93.4 | 75.8 | 79.2 | 76.8 |
| Max | 101.3 | 80.3 | 83.8 | 81.5 |
| 50 Min | 156.8 | 134.9 | 135.4 | 135.1 |
| Mean | 161.7 | 136.2 | 137.9 | 137.2 |
| Max | 164.1 | 137.9 | 140.5 | 138.2 |
| 100 Min | 197.6 | 181.4 | 182.1 | 182.7 |
| Mean | 199.3 | 182.5 | 183.2 | 183.4 |
| Max | 202.3 | 183.5 | 183.9 | 184.5 |

| Table 3 Simulation results (convergence). |
|---|---|---|---|---|
| N200 | LA | PLA | PLA' | SA |
| 200 10 | 83.1 | 105.8 | 6.6 | 266.2 |
| 20 | 95.1 | 134.6 | 7.9 | 306.4 |
| 50 | 105.6 | 332.2 | 15.8 | 347.4 |
| 100 | 131.5 | 1062.5 | 55.4 | 440.0 |
| 500 10 | 756.6 | 812.4 | 92.1 | 1139.5 |
| 20 | 773.3 | 884.7 | 115.1 | 1263.6 |
| 50 | 815.4 | 1165.5 | 151.2 | 1508.9 |
| 100 | 845.3 | 1809.8 | 207.1 | 1938.0 |
5. Conclusion

In present paper, as one of combinatorial optimization problems, the graph-partitioning problems was handled. Targeting at the parallelism and solution preciseness, we proposed a parallel learning automaton computing method with local evaluation function for the GPP and experimental verified the method.

Computer experiments showed the efficacy of the proposed method. The preciseness of the proposed method is the same or above level over the learning automaton computing method with local evaluation method, and is very close to that of simulated annealing. In the aspect of convergence time, the convergence is much faster to reach to certain preciseness.

As a direction of future research, constructing new reinforcement scheme and more effective algorithm to improve the learning preciseness and learning speed and constructing more effective data communication on clusters can be considered.

References


